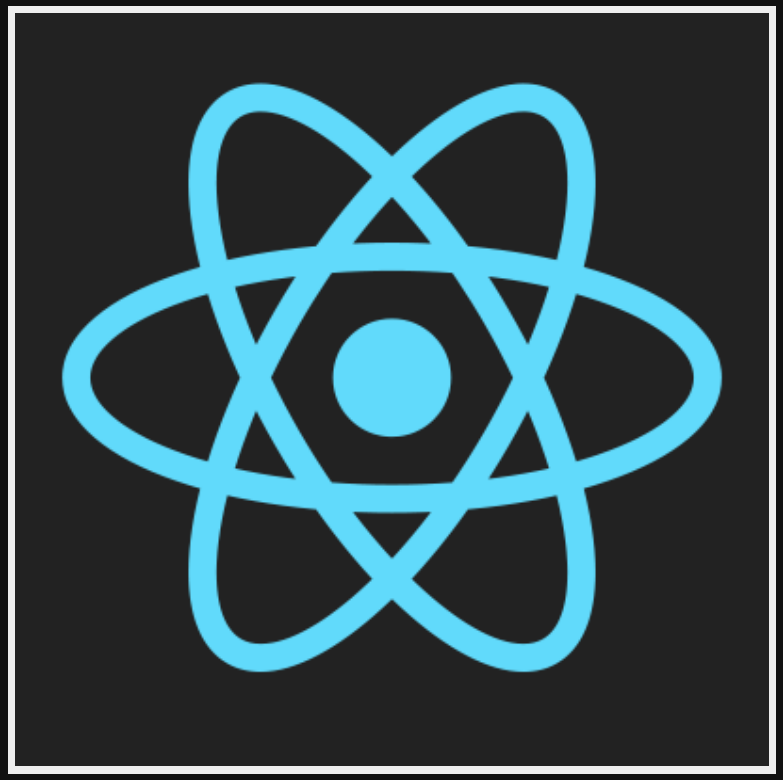


ReactJS

by

James Hrisho

About Me

Product Developer

Maxwell Health

- Github: securingsincity
- Twitter: @securingsincity
- http://www.jameshrisho.com

React

- Made By Facebook
- Released and Open Sourced May 2013

What React is not

Not a framework (Ember)

Not a framework for frameworks (Backbone,AngularJS)

What React is

"The V in MVC"

- The merging of DOM generation and display logic
- Components have DOM elements AND logic!
- Reusable components to create complex and large scale UI
- Only 28kb
- Support back to IE8

How Does React Work?

Most people will use JSX transformer to take this:

```
/** @jsx React.DOM */
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

React.renderComponent(<HelloMessage name="John" />, mountNode);
```

And turn it into this:

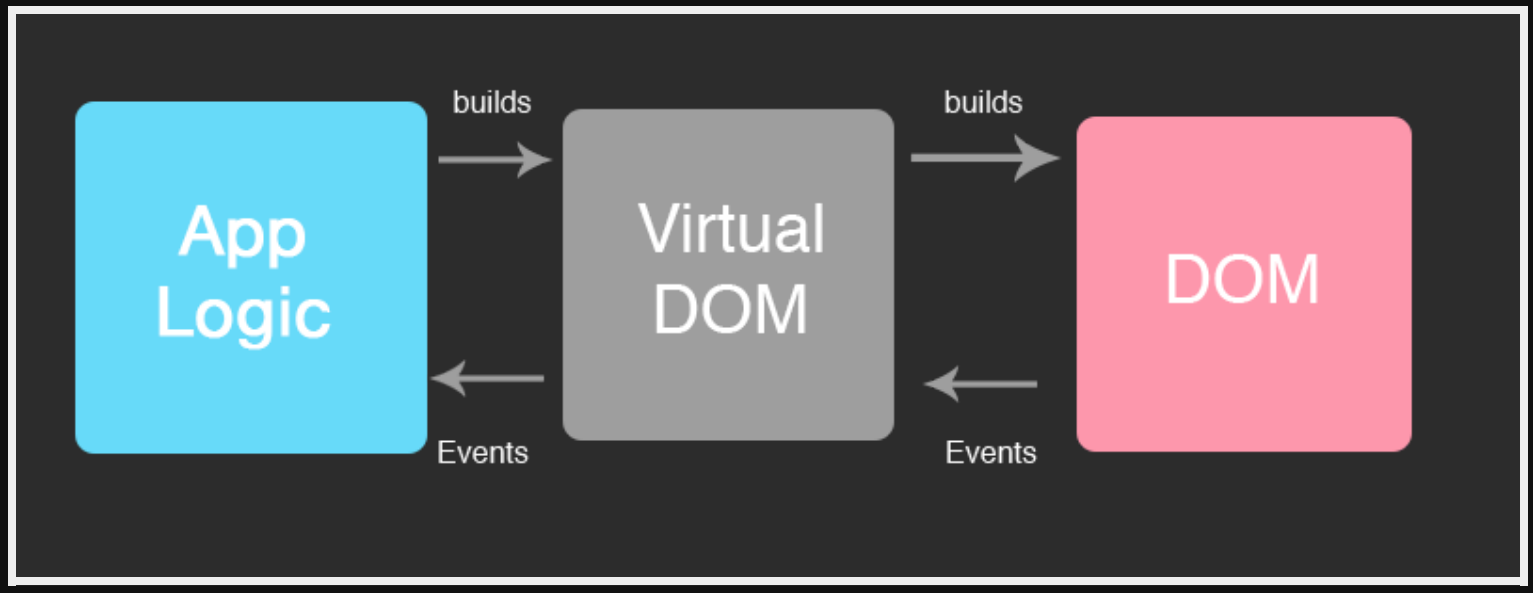
```
/** @jsx React.DOM */
var HelloMessage = React.createClass({displayName: 'HelloMessage',
  render: function() {
    return React.DOM.div(null, "Hello ", this.props.name);
  }
});

React.renderComponent(HelloMessage( {name:"John"} ), mountNode);
```

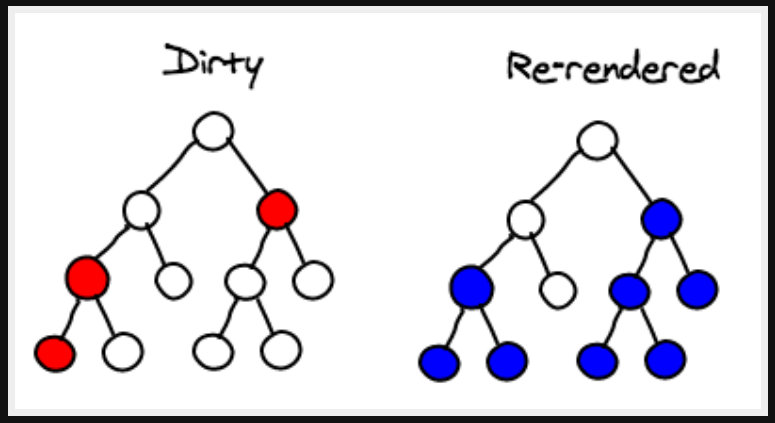
How Does React Work?

- full re-renders component to maintain state
  - no magical data-binding
  - no dirty model checking
  - no more explicit dom operations

How Does React Work?



How Does React Work?



- Every update:
  - New virtual dom subtree,
  - diffs it with the old subtree,
  - calculates only the most minimal changes
  - queues it up
  - and then batch executes

Getting Started

So what's the most basic example?

```
<html>
  <head>
    <script src="build/react.js" />
    <script src="build/JSXTransformer.js" />
  </head>
  <body>
    <div id="example"></div>
    <script type="text/jsx">
      /** @jsx React.DOM */
      React.renderComponent(
        <h1>Hello, world!</h1>,
        document.getElementById('example')
      );
    < /script>
  </body>
</html>
```

A Little Less Basic

State & Props Object

```
/** @jsx React.DOM */
var React = require('react');
var ClickMeButton = React.createClass({
   getInitialState: function() {
    return {count: 0};
  },
  oClick: function(e) {
    var count = this.state.count+ 1;
    this.setState({count: count});
  },
  render: function () {
    return (
      <div>
        <button onClick={this.oClick}>{this.props.name}</button> Count : {this.state.count}
      </div>
    );
  }
});

React.renderComponent(<ClickMeButton name='Click Me'/>, document.getElementById("content"));
```

Using React

- Running within a more complex stack
  - Gulp/Grunt for tasks
  - Browserify with Reactify

Gulpfile.js

```
  gulp.task('scripts', function () {
    return  browserify({
      entries: ['./app/js/main.jsx']
    })
    .bundle({debug:true})
    .pipe(source('app.js'))
    .pipe(gulp.dest('app/js'))
});
```

Package.json

```
  "browserify": {
    "transform": [
      "reactify"
    ]
  },
```

Testing React

Jest

- Run tests in a Virtual DOM
- Based on Jasmine
- Automatic Mocking
  - Implements its own version of `require()` to do the mocking
- If you are doing the things in the last slide
- You can achieve coverage very quickly

The best part? You can use it on anything. It makes testing `window` DOM elements

Testing React

Jest

```
var React = require('react/addons');
var gronkButton = require('../js/button.jsx');
var TestUtils = React.addons.TestUtils;
```

```
describe('button test', function() {
  it('changes the text after multiple clicks', function() {

    var button = <gronkButton name="hi"/>;
    TestUtils.renderIntoDocument(button);
    var div = TestUtils.findRenderedDOMComponentWithTag( button, 'div');
    var buttonDom = TestUtils.findRenderedDOMComponentWithTag(button, 'button');
    expect(div.getDOMNode().textContent).toEqual('hi Count : 0');
    for(var i = 1; i < 10; i++){
        React.addons.TestUtils.Simulate.click(buttonDom.getDOMNode());
        expect(div.getDOMNode().textContent).toEqual('hi Count : '+i);
    }
  });
});
```

Putting it all together

- Multiple reusable components
- Jest Tests
- Gulp and Browserify

http://github.com/securingsincity/react-jest-example

When can I start?

NOW

- In the browser on a super simple project just add a couple `<script>` tags
- More advanced projects with multiple components
- Pre-rendered using node served from the server
- With Backbone !!

(this is why Backbone and React are great!)

Just vanilla React and Backbone

https://github.com/jhudson8/react-backbone react-backbone mixins

More Info

React Docs:

http://facebook.github.io/react/

Jest Docs:

http://facebook.github.io/jest/

More from me:

http://github.com/securingsincity

Credits And Other Great Info:

React-backbone

https://github.com/jhudson8/react-backbone

Rethinking Best Practices by Pete Hunt

https://www.youtube.com/watch?v=x7cQ3mrcKaY

React's diff algorithm by Christopher Chedeau

http://calendar.perfplanet.com/2013/diff/